NORTH-HOLLAND

UPDATE BY MEANS OF INFERENCE RULES

TEODOR C. PRZYMUSINSKI AND HUDSON TURNER

▷ Katsuno and Mendelzon have distinguished two abstract frameworks for reasoning about change: theory revision and theory update. *Theory revision* involves a change in knowledge or belief with respect to a static world. By contrast, *theory update* involves a change of knowledge or belief in a changing world. In this paper, we are concerned with theory update. Winslett has shown that theory update should be computed "one model at a time." Accordingly, we focus exclusively on the update of interpretations. We begin with a study of *revision programming*, introduced by Marek and Truszczyński to formalize interpretation update in a language similar to logic programming. While revision programs provide a useful and natural definition of interpretation update, they are limited to a fairly restricted set of update rules. Accordingly, we introduce the more general notion of *rule update*—interpretation update by arbitrary sets of inference rules. We show that Winslett's approach to update by means of arbitrary sets of formulas corresponds to a simple subclass of rule update. We also specify a simple embedding of rule update in Reiter's *default logic*, obtained by augmenting the original update rules with default rules encoding the *commonsense law of inertia*—the principle that things change only when they are made to. © *Elsevier Science Inc., 1997* ◁

1. INTRODUCTION

Katsuno and Mendelzon [5] have distinguished two abstract frameworks for reasoning about change: theory revision and theory update. *Theory revision* involves a

Address correspondence to Teodor Przymusinski, Department of Computer Science, University of California, Riverside, CA 92521, E-mail: teodor@cs.ucr.edu; Hudson Turner, Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712, E-mail: hudson@cs.utexas.edu.

Received August 1995; accepted June 1996.

THE JOURNAL OF LOGIC PROGRAMMING

655 Avenue of the Americas, New York, NY 10010
0743-1066/97/$17.00
PII S0743-1066(96)00091-X

change in knowledge or belief with respect to a static world. For example, suppose you are booked on a flight, but told only that your destination is either Australia or Europe, i.e., that *australia* $\vee$ *europe* holds. If sometime later you learn, in addition to what you already know, that you were not booked on a flight to Europe, i.e., that $\neg$*europe* holds, then you are likely to conclude that your destination is Australia, i.e., that *australia* holds.

By contrast, *theory update* involves a change of knowledge or belief in a changing world. Again suppose you are booked on a flight, and told only that your destination is either Australia or Europe, i.e., that *australia* $\vee$ *europe* holds. Suppose you later learn that the situation has changed and all flights to Europe have just been cancelled, i.e., that $\neg$*europe* holds. Under these circumstances, you are not likely to conclude that you are going to Australia, i.e., that *australia* holds. In fact, it may be the case that your flight has just been cancelled.

In this paper, we are concerned with theory update. A key insight into the nature of update is due to Winslett [15], who showed that reasoning about actions should be done "one model at a time." That is, when reasoning about the outcome of an action, we must consider its effect in each one of the states of the world that are consistent with our (possibly incomplete) knowledge of the current state of the world. This insight is reflected in the general definition of theory update due to Katsuno and Mendelzon, which can be formulated as follows. Let Γ and T be sets of propositional formulas. A set T' of formulas is a "theory update" of T by Γ if

$$Models(T') = \{I' : \exists I \in Models(T) \cdot I' \text{ is "an update of } I \text{ by } \Gamma\text{"}\}.$$

We see by the form of this definition that in order to determine "theory update," it suffices to define when an interpretation I' is an update of an interpretation I by a theory Γ. Accordingly, in this paper, we focus exclusively on "interpretation update." However, in contrast to the work cited above, we investigate a more general case of update by means of sets $\mathcal{R}$ of inference rules, instead of sets Γ of formulas.

The first part of the paper is devoted to the study of *revision programs*, introduced by Marek and Truszczyński in a series of recent papers [6–8] to formalize interpretation update in a language similar to the language of logic programming. Revision programs are essentially sets of positive logic program rules, which can be interpreted as inference rules and used to update interpretations. Marek and Truszczyński proved that *logic programs with stable semantics* are embeddable into revision programs. We show that, conversely, there is a simple embedding of revision programs into logic programs with stable semantics.[1] Thus, the two formalisms are precisely equivalent. We go on to demonstrate that various properties of revision programs are easily derived from this translation and from known properties of logic programs.

Our translation of revision programs into logic programs utilizes a simple and intuitive encoding of the *commonsense law of inertia*, which is the principle that things do not change unless they are made to. The fact that revision programming is easily captured in logic programs using such inertia rules helps clarify the nature of revision programming, and as we will see, of interpretation update more generally.

While revision programs provide a useful and natural definition of interpretation update, they are limited to a fairly restricted set of update rules, and thus are

[1] Chitta Baral [1] independently found a somewhat more complex embedding.

not sufficiently expressive to capture more complex interpretation updates which may be described by arbitrarily complex formulas or, more generally, by arbitrary inference rules. Accordingly, in the second part of the paper, we introduce the notion of *rule update*—interpretation update by arbitrary sets of inference rules. The proposed formalism is not only more general and expressive than revision programming, but also has a very simple and natural definition.

We show that Winslett's [15] approach to update by means of arbitrary sets of formulas corresponds to a simple subclass of rule update. We also investigate how the "directionality" of inference rules contributes to the expressiveness of rule update. Finally, we specify a simple embedding of rule update into *default logic* [12], obtained by augmenting the original update rules with inertia axioms analogous to those used in the translation of revision programs into logic programs. The translation into default logic provides a bridge between our newly introduced formalism and a well-known nonmonotonic formalism.

The introduction of rule update provides a new framework for *interpretation update*, and thus also for *theory update*. In spite of its great simplicity, rule update constitutes a powerful and expressive mechanism which can be used to determine updates of theories by arbitrarily complex sets of inference rules and is applicable to various knowledge domains. For example, in [9], McCain and Turner apply rule update to the problem of reasoning about the effects of actions. Moreover, the simple embedding of rule update in default logic—obtained by adding default rules encoding the commonsense law of inertia—provides a crucial element of proposals in [13, 14] for representing commonsense knowledge about actions in default logic and logic programming.

Preliminary definitions appear in Section 2. In Sections 3 and 4, we specify a simple embedding of revision programming in logic programming, and show that basic results obtained by Marek and Truszczyński for revision programs are easily deduced from this embedding, using known properties of logic programs. In Section 5, we introduce an alternative characterization of revision programming, and in Section 6, we define the more general notion of rule update as a natural extension of this alternative characterization. In Section 7, we compare rule update to Winslett's [15] definition of update by means of propositional formulas, and we investigate how the "directionality" of inference rules influences rule update. In Section 8, we specify an embedding of rule update in default logic. Section 9 consists of a few concluding remarks.

2. PRELIMINARY DEFINITIONS

Let $\mathcal{K}$ be a propositional language. For any set Γ of formulas from $\mathcal{K}$, by $Cn(\Gamma)$ we denote the least logically closed set of formulas from $\mathcal{K}$ that contains Γ. Inference rules over $\mathcal{K}$ will be written as expressions of the form

$$\frac{\phi}{\psi}$$

where ϕ and ψ are formulas from $\mathcal{K}$. We often find it convenient to identify a propositional formula ϕ with the inference rule

$$\frac{True}{\phi}.$$

Let $\mathcal{R}$ be a set of inference rules over $\mathcal{K}$, and let Γ be a set of formulas from $\mathcal{K}$. We say Γ is *closed under* $\mathcal{R}$ if for every rule $\phi/\psi \in \mathcal{R}$, if ϕ belongs to Γ, then ψ does too. We write

$$\Gamma \vdash_{\mathcal{R}} \phi$$

if ϕ belongs to the least logically closed set of formulas from $\mathcal{K}$ that contains Γ and is closed under $\mathcal{R}$.

A *default rule* over $\mathcal{K}$ is an expression of the form

$$\frac{\alpha\colon \beta_1, \ldots, \beta_n}{\gamma} \tag{2.1}$$

where all of $\alpha, \beta_1, \ldots, \beta_n, \gamma$ are formulas from $\mathcal{K}$. For a default rule r as in (2.1), we define $\mathit{prerequisite}(r) = \alpha$, $\mathit{justifications}(r) = \{\beta_1, \ldots, \beta_n\}$, and $\mathit{consequent}(r) = \gamma$. If $\mathit{prerequisite}(r) = \mathit{True}$, we sometimes omit it and write $: \beta_1, \ldots, \beta_n/\gamma$ instead. If $\mathit{justifications}(r)$ is empty, we identify r with the corresponding inference rule α/γ. If, in addition, $\mathit{prerequisite}(r) = \mathit{True}$, we sometimes simply write γ.

A *default theory* over $\mathcal{K}$ is a set of default rules over $\mathcal{K}$. Let D be a default theory over $\mathcal{K}$, and let E be a set of formulas from $\mathcal{K}$. We define the *reduct of D with respect to E*, denoted by D^E, as follows.

$$D^E = \left\{ \frac{\mathit{prerequisite}(r)}{\mathit{consequent}(r)} : \qquad r \in D \text{ and for all } \beta \in \mathit{justifications}(r), \neg\beta \notin E \right\}$$

We say that E is an *extension* of D if E is the least logically closed set that is closed under D^E.[2]

A *logic program* over $\mathcal{K}$ is a set of *logic program rules* over $\mathcal{K}$, which are expressions of the form

$$A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n$$

where A, B_i, and C_j are atoms from $\mathcal{K}$ $(m, n \geq 0)$. If $n = 0$ for all program rules, then the program is called *positive*. By interpreting lists of atoms in rule bodies as conjunctions of atoms, we can identify a positive logic program with a propositional Horn theory.[3]

Definition 2.1 (Stable models) [2]. Let P be a logic program over a language $\mathcal{K}$, and let M be an interpretation of $\mathcal{K}$. By the *quotient of P modulo M*, we mean the positive logic program P/M obtained from P by:

- removing from P all rules which contain a negative premise "not C" such that C is true in M, and
- deleting all negative premises "not C" from the remaining rules of P.

Since the program P/M is a Horn theory, it has a unique least model $\mathit{Least}(P/M)$. The interpretation M is called a *stable model* of the program P if $M = \mathit{Least}(P/M)$.

[2]Default logic is due to Reiter [12]. The definition of an extension given above follows [4], and is equivalent to Reiter's definition.

[3]Before continuing, we recall the fact that propositional programs and default theories can be viewed as instantiated versions of programs and (quantifier-free) theories with variables. Thus, the results in this paper apply to the general case.

Definition 2.2 (Extended logic programs) [3] (see also [11]). Let $\mathcal{K}$ be a propositional language. Let $\mathcal{K}^*$ be an *extended propositional language* obtained by augmenting $\mathcal{K}$ with new propositional letters $\sim A$, for some (or all) propositional letters A in $\mathcal{K}$. The new propositional symbols $\sim A$ are called *strong* (or "classical") negation of A. A logic program P over the extended language $\mathcal{K}^*$ is called an *extended logic program*. A stable model of P over $\mathcal{K}^*$ is an *extended stable model* of P if there is no atom $A \in \mathcal{K}$ such that both A and $\sim A$ are true in M.

3. EMBEDDING REVISION PROGRAMS INTO LOGIC PROGRAMS

Revision programs were introduced by Marek and Truszczyński in a series of papers [6–8] in order to formalize interpretation update in a language similar to the language of logic programming. In [6], they showed that logic programs with stable semantics are embeddable into revision programs. In this section, we specify a remarkably simple embedding of revision programs into logic programs with stable semantics. Consequently, the two formalisms are precisely equivalent. In the next section, we demonstrate how one can easily derive various properties of revision programs from this translation and from known properties of logic programs.

3.1. Revision Programs

We first recall the definition of revision programs. Following [7], we fix a countable set U.

Definition 3.1 (Revision programs) [7]. A *revision in-rule* or, simply, an *in-rule*, is any expression of the form

$$in(p) \leftarrow in(q_1), \ldots, in(q_m), out(s_1), \ldots, out(s_n) \tag{3.1}$$

where p, q_i, $1 \leq i \leq m$, and s_j, $1 \leq j \leq n$, are all in U and $m, n \geq 0$. A *revision out-rule* or, simply, an *out-rule*, is any expression of the form

$$out(p) \leftarrow in(q_1), \ldots, in(q_m), out(s_1), \ldots, out(s_n) \tag{3.2}$$

where p, q_i, $1 \leq i \leq m$, and s_j, $1 \leq j \leq n$, are all in U and $m, n \geq 0$. A collection of in-rules and out-rules is called a *revision program*. Any subset B of U is called a *knowledge base*.

Clearly, revision programs can be syntactically viewed as positive logic programs (or as propositional Horn theories). However, as we will see below, they are given a special revision semantics which differs significantly from the least model semantics of positive logic programs. We first need the definition of the necessary change determined by a revision program.

Definition 3.2 (Necessary change) [7]. Let P be a revision program with least model M. The *necessary change determined by* P is the pair (I, O), where $I = \{q\colon in(q) \in M\}$ and $O = \{q\colon out(q) \in M\}$. The revision program is called *coherent* if $I \cap O = \emptyset$.

Now we are ready to define the so-called P-justified revisions.

Definition 3.3 (P-justified revision) [7].[4] Suppose that P is a revision program, B_I is the initial knowledge base, and B_R is the revised knowledge base. The *reduct of P with respect to (B_I, B_R)* is defined as the revision program $P_{B_R} \mid B_I$ obtained from P by:

- removing from the body of each rule in P all atoms $in(a)$ such that $a \in B_I \cap B_R$, and all atoms $out(a)$ such that $a \notin B_I \cup B_R$;
- removing every rule of type (3.1) or (3.2) such that $q_i \notin B_R$, for some i, $1 \leq i \leq m$, or $s_j \in B_R$, for some j, $1 \leq j \leq n$.

If (I, O) is the necessary change determined by $P_{B_R} \mid B_I$ and $P_{B_R} \mid B_I$ is coherent and $B_R = (B_I \cup I) - O$, then B_R is called a *P-justified revision* of B_I.

The reader is referred to [7] for the motivation of the above definition. The results established in the remainder of the paper, e.g., Theorem 3.1, will further clarify its intuitive meaning and its relationship to other well-established notions.

According to the following lemma, the last condition $B_R = (B_I \cup I) - O$ can be equivalently stated as $B_R = (B_R \cap B_I) \cup I$ and $\overline{B}_R = (\overline{B}_R \cap \overline{B}_I) \cup O$, where by $\overline{A}$ we denote the complement $U - A$ of the set A.

Lemma 3.1. Suppose that B_I, B_R, O, I are subsets of U such that $O \cap I = \emptyset$. The following two conditions are equivalent:

(i) $B_R = (B_I \cup I) - O$;
(ii) $B_R = (B_R \cap B_I) \cup I$ *and* $\overline{B}_R = (\overline{B}_R \cap \overline{B}_I) \cup O$.

PROOF. Suppose that (ii) holds. We have to show that $B_R = (B_I - O) \cup I$. Clearly, (ii) implies that $B_R \supseteq I$. Moreover, if $q \in B_I - O$, then q does not belong to $\overline{B}_R$, and therefore $q \in B_R$. This shows that $B_R \supseteq (B_I - O) \cup I$.

If $q \in B_R$, then, by (ii), $q \in B_I \cup I$. Moreover, $q \notin O$, which shows that $B_R \subseteq (B_I - O) \cup I$, and therefore $B_R = (B_I - O) \cup I = (B_I \cup I) - O$.

Suppose now that (i) holds. We have to show that $B_R = (B_R \cap B_I) \cup I$ and $\overline{B}_R = (\overline{B}_R \cap \overline{B}_I) \cup O$. Clearly, by (i), $B_R \supseteq (B_R \cap B_I) \cup I$. Moreover, if $q \in B_R$, then either $q \in I$ or $q \in B_I$, which shows that $B_R \subseteq (B_R \cap B_I) \cup I$, and therefore $B_R = (B_R \cap B_I) \cup I$.

Since O is disjoint from B_R, we infer that $\overline{B}_R \supseteq (\overline{B}_R \cap \overline{B}_I) \cup O$. Moreover, if $q \in \overline{B}_R$ and $q \notin O$, then $q \notin B_I$, which shows that $\overline{B}_R \subseteq (\overline{B}_R \cap \overline{B}_I) \cup O$, and therefore $\overline{B}_R = (\overline{B}_R \cap \overline{B}_I) \cup O$. □

3.2. Translating Revision Programs into Logic Programs

We next show how to embed revision programs into logic programs with stable semantics. We employ a propositional language $\mathcal{K}$ whose set of propositional letters is $\{in(q) : q \in U\} \cup \{out(q) : q \in U\} \cup \{in_I(q) : q \in U\} \cup \{out_I(q) : q \in U\}$.

Definition 3.4 (Translating revision programs into logic programs). The translation of a revision program P and an initial knowledge base B_I into a logic program

[4] Although the definition given here differs slightly from the one given in [7], it is easily seen to be equivalent.

is defined as the logic program $\mathcal{P}(P, B_I) = P_I \cup P_N \cup P$ over $\mathcal{K}$ consisting of the following three subprograms.

Initial Knowledge Rules P_I: All $q \in B_I$ are initially in and all $s \notin B_I$ are initially out:

$$in_I(q) \leftarrow \tag{3.3}$$

$$out_I(s) \leftarrow \tag{3.4}$$

for all $q \in B_I$ and all $s \notin B_I$.

Inertia Rules P_N: If q was initially in (respectively, out), then after revision it remains in (respectively, out) unless it was forced out (respectively, in):

$$in(q) \leftarrow in_I(q), \text{not } out(q) \tag{3.5}$$

$$out(q) \leftarrow out_I(q), \text{not } in(q) \tag{3.6}$$

for all $q \in U$.

Revision Rules P: All the in-rules and out-rules that belong to the original revision program P.

A stable model M of $\mathcal{P}(P, B_I)$ is called *coherent* if it does not contain both $in(q)$ and $out(q)$, for any $q \in U$.

Observe that the above translation is quite simple. It preserves the original revision program P, and adds to it the set of facts representing the initial state B_I and two simple inertia axiom schemas stating that things do not change from one state to another unless they are forced to.[5]

Example 3.1. Consider the revision program

$$P = \{out(a) \leftarrow in(b)\}$$

and the initial knowledge base $B_I = \{a,\ b\}$. Its translation $\mathcal{P}(P, B_I)$ into a logic program consists of P together with the initial conditions and inertia axioms[6]:

$$in_I(a)$$

$$in_I(b)$$

$$in(a) \leftarrow in_I(a), \text{not } out(a)$$

$$in(b) \leftarrow in_I(b), \text{not } out(b).$$

One easily checks that program $\mathcal{P}(P, B_I)$ has a unique stable model, which contains only the atoms[7]:

$$\{out(a),\ in(b)\}$$

and therefore corresponds to the unique P-justified revision $B_R = \{b\}$.

[5] By comparison, the translation in [1] is complicated by the introduction of an auxiliary abnormality predicate.

[6] Notice that the inertia axioms for *out* can be skipped in this case.

[7] In addition to the initial state atoms $in_I(a)$, $in_I(b)$.

Example 3.2. Consider now the revision program P:

$$in(a) \leftarrow out(b)$$
$$in(b) \leftarrow out(a)$$

and the initial knowledge base $B_I = \{\}$. Its translation $\mathcal{P}(P, B_I)$ into a logic program consists of P together with the initial conditions and inertia axioms[8]:

$$out_I(a)$$
$$out_I(b)$$
$$out(a) \leftarrow out_I(a), \text{not } in(a)$$
$$out(b) \leftarrow out_I(b), \text{not } in(b).$$

One easily checks that program $\mathcal{P}(P, B_I)$ has two stable models M_1 and M_2, which contain only the atoms[9]:

$$M_1 = \{out(a),\ in(b)\}$$
$$M_2 = \{in(a),\ out(b)\}$$

and therefore correspond to the two P-justified revisions $\{b\}$ and $\{a\}$.

We now prove that the translation specified in Definition 3.4 indeed yields an embedding of revision programming into logic programming under the stable semantics.

Theorem 3.1 (Embedding revision programming in logic programming). *Let P be a revision program, and let B_I be an initial knowledge base. There is a one-to-one correspondence between P-justified revisions of B_I and coherent stable models of its translation $\mathcal{P}(P, B_I)$ into a logic program.*

More precisely, to every P-justified revision B_R of B_I there corresponds a unique stable model M of $\mathcal{P}(P, B_I)$ such that

$$B_R = \{q\colon in(q) \in M\} \tag{3.7}$$
$$U - B_R = \{q\colon out(q) \in M\} \tag{3.8}$$

and, conversely, for every coherent stable model M of $\mathcal{P}(P, B_I)$, the set $B_R = \{q\colon in(q) \in M\}$ is a P-justified revision of B_I.

PROOF. ($\Leftarrow$) Suppose that M is a coherent stable model of $\mathcal{P}(P, B_I)$, and let $B_R = \{q\colon in(q) \in M\}$. We must show that B_R is a P-justified revision of B_I.

By Definition 2.1, M is the least model of the positive logic program $Q = \mathcal{P}(P, B_I)/M$, namely, the quotient of $\mathcal{P}(P, B_I)$ modulo M. Since both the initial knowledge rules P_I and the original revision rules P in $\mathcal{P}(P, B_I) = P_I \cup P_N \cup P$ are positive, only the inertia rules P_N

$$in(q) \leftarrow in_I(q), \text{not } out(q) \tag{3.9}$$
$$out(q) \leftarrow out_I(q), \text{not } in(q) \tag{3.10}$$

will be affected by the quotient transformation, and therefore $Q = P_I \cup (P_N/M) \cup P$.

[8]Notice that the inertia axioms for *in* can be skipped in this case.

[9]In addition to the initial state atoms $out_I(a)$, $out_I(b)$.

Define $B'_R = \{q\colon out(q) \in M\}$, and let $\overline{B}_I = U - B_I$. According to Definition 2.1, in order to construct the quotient P_N/M, we have to remove from P_N all the inertia clauses (3.9) such that $q \in B'_R$ and all inertia clauses (3.10) such that $q \in B_R$. Subsequently, we have to remove all the negative premises from the remaining clauses of P_N. As a result of the quotient transformation, we therefore obtain the program P_N/M consisting of rules

$$in(q) \leftarrow in_I(q), \qquad \text{for all } q \notin B'_R \tag{3.11}$$

$$out(q) \leftarrow out_I(q), \qquad \text{for all } q \notin B_R. \tag{3.12}$$

Let us now observe that $B'_R = \overline{B}_R = U - B_R$. Indeed, since M is coherent, the sets B_R and B'_R are disjoint. Suppose that there is a q such that $q \notin B_R$ and $q \notin B'_R$. Then both clauses $in(q) \leftarrow in_I(q)$ and $out(q) \leftarrow out_I(q)$ belong to the quotient program Q. Since we must either have $q \in B_I$ or $q \in \overline{B}_I$ and since M is the least model of Q, we conclude that either $in(q)$ or $out(q)$ must belong to M, which contradicts our assumption that $q \notin B_R$ and $q \notin B'_R$.

Clearly, M is also the least model of a modified program obtained by removing some premises which are true in M. Therefore, we can further reduce the quotient P_N/M of the set of inertia rules to the set $\widehat{P}_N/M$ of all clauses (facts) of the form

$$in(q) \leftarrow , \qquad \text{for all } q \in B_R \cap B_I \tag{3.13}$$

$$out(q) \leftarrow , \qquad \text{for all } q \in \overline{B}_R \cap \overline{B}_I. \tag{3.14}$$

In addition, the quotient program Q contains the initial knowledge rules P_I:

$$in_I(q) \leftarrow , \qquad \text{for all } q \in B_I \tag{3.15}$$

$$out_I(q) \leftarrow , \qquad \text{for all } q \in \overline{B}_I \tag{3.16}$$

and all the original revision program in-rules (3.1) and out-rules (3.2) in P.

We now show that B_R is a P-justified revision of B_I. According to Definition 3.1, in order to compute the reduct $P_{B_R} \mid B_I$ of P, we first have to remove from the body of each revision rule in P all atoms $in(q)$ such that $q \in B_I \cap B_R$, and all atoms $out(s)$ such that $s \in \overline{B}_I \cap \overline{B}_R$. Notice that these are precisely the atoms that must be true in M due to the rules (3.13) and (3.14). Subsequently, we remove from the (already reduced) revision program every rule of type (3.1) or (3.2) such that $q_i \notin B_R$, for some i, $1 \leq i \leq m$, or $s_j \in B_R$, for some j, $1 \leq j \leq n$, thus obtaining the reduct $P_{B_R} \mid B_I$. Notice that by doing so, we are removing from P those rules whose premises are false in M. As a result, the stable model M remains the least model of the reduced quotient program $Q^* = P_I \cup (\widehat{P}_N/M) \cup P_{B_R} \mid B_I$.

Let M_0 be the least model of the reduct $P_{B_R} \mid B_I$, and let (I, O) be the necessary change of $P_{B_R} \mid B_I$, i.e., $I = \{q\colon in(q) \in M_0\}$ and $O = \{q\colon out(q) \in M_0\}$. The program $Q^* = P_I \cup (\widehat{P}_N/M) \cup P_{B_R} \mid B_I$ consists of three independent parts: the initial knowledge rules (3.15) and (3.16), the (reduced) inertia axioms (3.13) and (3.14), and the reduct $P_{B_R} \mid B_I$ which no longer contains any premises from the other two parts. Consequently, the set of atoms that belong to the stable model M, which is the least model of this reduced quotient program Q^*, consists of:

- $\{in_I(q) : q \in B_I\} \cup \{out_I(q) : q \in \overline{B}_I\}$,
- $\{in(q) : q \in B_R \cap B_I\} \cup \{out(q) : q \in \overline{B}_R \cap \overline{B}_I\}$,
- $\{in(q) : q \in I\} \cup \{out(q) : q \in O\}$.

This shows that

$$B_R = \{q\colon in(q) \in M\} = (B_R \cap B_I) \cup I, \tag{3.17}$$

$$\overline{B}_R = U - B_R = \{q\colon out(q) \in M\} = (\overline{B}_R \cap \overline{B}_I) \cup O. \tag{3.18}$$

Since $P_{B_R} \mid B_I$ is coherent by assumption, in order to verify that B_R is a P-justified revision of B_I, it suffices to establish that $B_R = (B_I \cup I) - O$. However, this follows immediately from Lemma 3.1.

($\Rightarrow$) The proof in the opposite direction is very similar, and thus will be skipped. We begin with a P-justified revision B_R of B_I. From Lemma 3.1, we conclude that the conditions $B_R = (B_R \cap B_I) \cup I$ and $\overline{B}_R = (\overline{B}_R \cap \overline{B}_I) \cup O$ must be satisfied. Using this fact and reversing the steps of the above proof, we produce the required stable model M. □

4. PROPERTIES OF REVISION PROGRAMS

The embedding of revision programs into logic programs with stable semantics helps clarify the notion of revision programming, in part because it allows us to utilize our knowledge of an already well-established and thoroughly investigated nonmonotonic formalism. For instance, many of the results obtained by Marek and Truszczyński in [6–8] are simple consequences of the embedding. In this section, we illustrate this claim with a few examples.

We begin with the following result from [7] stating that logic programs with stable semantics are embeddable into revision programs.

Theorem 4.1 [7]. *Let P be a logic program, which consists of rules of the form*

$$p \leftarrow q_1, \ldots, q_m, \text{not } s_1, \ldots, \text{not } s_n$$

and let $R(P)$ be the revision program obtained by replacing each rule of P with the corresponding in-rule

$$in(p) \leftarrow in(q_1), \ldots, in(q_m), out(s_1), \ldots, out(s_n).$$

An interpretation M is a stable model of P if and only if its set of atoms B_R is an $R(P)$-justified revision of $B_I = \emptyset$.

PROOF. By Definition 3.4, the translation $\mathcal{P}(R(P), \emptyset)$ of the revision program $R(P)$ into a logic program consists of $R(P)$ itself and the initial knowledge rules $out_I(q) \leftarrow$ for all $q \in U$, together with the inertia rules[10]

$$out(q) \leftarrow \text{not } in(q), \qquad \text{for all } q \in U.$$

After performing a single step of partial evaluation on the premises $out(s_j)$ of rules from $R(P)$ (by using the above inertia rules), the rules of $R(P)$ become equivalent to

$$in(p) \leftarrow in(q_1), \ldots, in(q_m), \text{not } in(s_1), \ldots, \text{not } in(s_n)$$

[10]Notice that we can skip the rules for in and remove the premises $out_I(q)$.

and thus they are equivalent (up to renaming) to the rules of the original program P. Theorem 3.1 now easily implies the equivalence between stable models of P and $R(P)$-justified revisions of B_I. □

In light of the embedding of revision programming into logic programming (Theorem 3.1), this result shows that the two formalisms are in fact precisely equivalent. Given this equivalence, the complexity results obtained in [8, Theorem 4.2], regarding the NP-completeness of some problems involving the computation of P-justified revisions, can be easily seen to follow from similar results already known about the computation of stable models.

Finally, the fact that the translation $\mathcal{P}(P, B_I)$ of a revision program is completely symmetric with respect to *in* and *out* atoms immediately yields the following result.

Theorem 4.2 [7]. *Let P be a revision program, and let B_I be a knowledge base. A knowledge base B_R is a P-justified revision of B_I if and only if $U - B_R$ is a P^D-justified revision of $U - B_I$, where P^D is the dual of the program P obtained by simultaneously replacing everywhere in by out and vice versa.*

5. ALTERNATIVE ACCOUNT OF REVISION PROGRAMMING

In this section, we consider an alternative embedding of revision programming in logic programming, which is somewhat more compact than the previous one (Definition 3.4), but otherwise very similar. This second embedding suggests a simple alternative characterization of revision programming, which serves as the basis for the definition of interpretation update by arbitrary sets of inference rules—or rule update—that is introduced in the next section.

For this alternative embedding, we use extended logic programs under the stable semantics (Definition 2.2), and we employ a more compact encoding of the commonsense law of inertia—the principle that things change only when made to.

Given the set U associated with a revision program P, let $\mathcal{K}^*$ be the propositional language with the atoms $U \cup \{\sim a : a \in U\}$. We will translate revision program P, along with an initial knowledge base B_I into an extended logic program over $\mathcal{K}^*$.

In this section, it will be convenient to represent interpretations of the extended language $\mathcal{K}^*$ as sets of atoms a and $\sim a$.

Definition 5.1 (Translation of revision programming into extended logic programming). Let P be a revision program, with initial knowledge base B_I. Let $\pi(P)$ be the positive extended logic program over $\mathcal{K}^*$ obtained from P by replacing each in-rule (3.1) in P with the corresponding rule

$$p \leftarrow q_1, \ldots, q_m, \sim s_1, \ldots, \sim s_n$$

and similarly replacing each out-rule (3.2) in P with the corresponding rule

$$\sim p \leftarrow q_1, \ldots, q_m, \sim s_1, \ldots, \sim s_n.$$

Let $\mathcal{P}^*(P, B_I)$ be the extended logic program over $\mathcal{K}^*$ that is obtained by augmenting $\pi(P)$ with the rule

$$a \leftarrow \text{not } \sim a \tag{5.1}$$

for each atom $a \in B_I$ and the rule

$$\sim a \leftarrow \text{not } a \tag{5.2}$$

for each atom $a \in U \setminus B_I$.

Unlike the first translation from revision programming to logic programming (Definition 3.4), this second translation does not include an explicit representation of the initial knowledge base B_I, but instead uses B_I to determine which rules of the forms (5.1) and (5.2) are to be added to the positive extended logic program $\pi(P)$. Intuitively speaking, rules of the forms (5.1) and (5.2) correspond to a partial evalution of the Inertia Rules [(3.5) and (3.6)] with respect to the Initial Knowledge Rules [(3.3) and (3.4)]. In this manner, we obtain a more compact translation which, nonetheless, still reflects the commonsense law of inertia.

Example 5.1. Consider again the revision program P and initial knowledge base B_I from Example 3.2. The program $\mathcal{P}^*(P, B_I)$ consists of the following four rules:

$$a \leftarrow \sim b$$

$$b \leftarrow \sim a$$

$$\sim a \leftarrow \text{not } a$$

$$\sim b \leftarrow \text{not } b.$$

Recall that the two P-justified revisions of B_I are the knowledge bases $\{a\}$ and $\{b\}$. Observe that these two P-justified revisions of B_I correspond precisely to the two extended stable models of $\mathcal{P}^*(P, B_I)$, which are $\{a, \sim b\}$ and $\{\sim a, b\}$.

In order to state the correctness of this second translation, we introduce the following auxiliary definition, relating knowledge bases and interpretations of $\mathcal{K}^*$ in the obvious way.

Definition 5.2. Let $\mathcal{M}$ be the injective function from the set of knowledge bases (that is, the set of subsets of U) to the set of interpretations of $\mathcal{K}^*$ such that, for any knowledge base B, $\mathcal{M}(B) = B \cup \{\sim a\colon a \in U \setminus B\}$.

Theorem 5.1 (Embedding revision programming in extended logic programming). *Let P be a revision program, with initial knowledge base B_I. A knowledge base B_R is a P-justified revision of B_I if and only if the interpretation $\mathcal{M}(B_R)$ is an extended stable model of $\mathcal{P}^*(P, B_I)$. Moreover, for every extended stable model M of $\mathcal{P}^*(P, B_I)$, there is a unique knowledge base B_R such that $M = \mathcal{M}(B_R)$.*

We do not include a proof of this theorem, which follows in straightforward fashion from the correctness of the first translation (Theorem 3.1).

Now, it is easy to verify that for any knowledge base B_R, we have

$$\frac{\mathcal{P}^*(P, B_I)}{\mathcal{M}(B_R)} = \pi(P) \cup (\mathcal{M}(B_I) \cap \mathcal{M}(B_R)).$$

This observation immediately yields the following corollary to Theorem 5.1.

Corollary 5.1 (Alternative characterization of revision programming). *Let P be a revision program, with initial knowledge base B_I. A knowledge base B_R is*

a P-justified revision of B_I if and only if $\mathcal{M}(B_R)$ is an extended stable model of $\pi(P) \cup (\mathcal{M}(B_I) \cap \mathcal{M}(B_R))$.

6. RULE UPDATE

Revision programs are sets of revision in-rules and out-rules, which can be interpreted as positive logic program rules, or as inference rules, and used to update interpretations. In this section, we introduce a more general approach to interpretation update, which allows update by means of arbitrary sets of inference rules. We call this more general notion rule update.

Rule update has a simple fixpoint definition which can be viewed as an extension of the alternative characterization of revision programming introduced in Corollary 5.1 of the previous section. Rule update not only extends revision programming, but also includes as a special case the approach to update by means of formulas introduced by Winslett in [15]. Furthermore, rule update has a simple embedding in default logic, using essentially the same inertia rules used in the previous section to embed revision programming in extended logic programming.

In the remaining sections of the paper, we will represent interpretations of a propositional language $\mathcal{K}$ as maximal consistent sets of literals from $\mathcal{K}$. This choice of representation allows us to characterize the set of facts that a pair I and I' of interpretations have in common simply by taking their intersection $I \cap I'$.

Definition 6.1 (Rule update). Let $\mathcal{R}$ be a set of inference rules. Let I and I' be interpretations. We say that I' is an *update of I by $\mathcal{R}$* if

$$I' = \{L\colon I \cap I' \vdash_{\mathcal{R}} L\}$$

where L ranges over literals.

The literals in $I \cap I'$ can be understood as the facts that are "preserved by inertia" as we move from interpretation I to interpretation I'. In accordance with the commonsense law of inertia, our definition of rule update does not require any additional "explanation" for the truth of these literals in I'. The definition does require, though, that all new facts in I'—that is, the literals in $I' \setminus I$—be explained by the rules in $\mathcal{R}$, along with the literals in $I \cap I'$. Accordingly, we see that I' is an update of I by $\mathcal{R}$ if and only if the following two conditions are met:

- for all literals L in $I' \setminus I$, $I \cap I' \vdash_{\mathcal{R}} L$;
- $Cn(I')$ is closed under $\mathcal{R}$.

That is, roughly speaking, I' must be "consistent with" the rules in $\mathcal{R}$, and every literal in I' must be explained—either it held in I or it was forced to become true.

Example 6.1. Consider the following:

$$I_1 = \{a, b, c\}, \quad \mathcal{R}_1 = \left\{\frac{a}{\neg b \vee \neg c}\right\}, \quad I_2 = \{a, \neg b, c\}.$$

First, we will show that I_2 is an update of I_1 by $\mathcal{R}_1$. Notice that $I_1 \cap I_2 = \{a, c\}$ and that $I_1 \cap I_2 \vdash_{\mathcal{R}_1} \neg b$. So for all literals $L \in I_2 \setminus I_1$, $I_1 \cap I_2 \vdash_{\mathcal{R}_1} L$. And since $Cn(I_2)$ is closed under $\mathcal{R}_1$, we have shown that I_2 is an update of I_1 by $\mathcal{R}_1$.

A symmetric argument shows that the interpretation $\{a, b, \neg c\}$ is also an update of I_1 by $\mathcal{R}_1$. On the other hand, if we take $I_3 = \{\neg a, b, c\}$, then $I_1 \cap I_3 = \{b, c\}$; and we see that $I_1 \cap I_3 \not\vdash_{\mathcal{R}_1} \neg a$. So I_3 is not an update of I_1 by $\mathcal{R}_1$. One can similarly show that the interpretation $\{a, \neg b, \neg c\}$ is not an update of I_1 by $\mathcal{R}_1$.

The following theorem, establishing that rule update indeed extends revision programming, is easily seen to follow from the alternative fixpoint characterization of revision programming captured in Corollary 5.1, in light of the strong resemblance between that characterization of revision programming and the definition of rule update.

Theorem 6.1 (Rule update subsumes revision programming). *Let P be a revision program, with initial and final knowledge bases B_I and B_R. Let $\mathcal{R}$ be the set of inference rules obtained by replacing each in-rule* (3.1) *in P with the corresponding inference rule*

$$\frac{q_1 \wedge \cdots \wedge q_m \wedge \neg s_1 \wedge \cdots \wedge \neg s_n}{p}$$

and similarly replacing each out-rule (3.2) *in P with the corresponding inference rule*

$$\frac{q_1 \wedge \cdots \wedge q_m \wedge \neg s_1 \wedge \cdots \wedge \neg s_n}{\neg p}.$$

Let I and I' be interpretations such that $I \cap U = B_I$ and $I' \cap U = B_R$. B_R is a P-justified revision of B_I if and only if I' is an update of I by $\mathcal{R}$.

7. PROPERTIES OF RULE UPDATE

In this section, we show that rule update includes as a special case the approach to update by means of formulas introduced by Winslett [15].[11] More generally, we briefly investigate how the "directionality" of inference rules contributes to the expressiveness of update by means of inference rules.

Definition 7.1 (Formula update).[12] Given interpretations I, I', and I'', we say that I' is *closer to I than I'' is* if $I'' \cap I$ is a proper subset of $I' \cap I$.

Let Γ be a set of formulas. Let I and I' be interpretations. We say that I' is a *formula-update of I by Γ* if I' is a model of Γ such that no model of Γ is closer to I than I' is.

The "principle of minimal change" that is transparently captured in this definition is closely related to the commonsense law of inertia that underlies rule update, as we will see. Intuitively speaking, the principle of minimal change stipulates that there be as few changes as possible, whereas the commonsense law of inertia assumes that things change only when made to.

[11] McCain and Turner [9] discuss this comparison at some length, in the framework of reasoning about action. Propositions 7.1–7.3 below are essentially identical to Propositions 2–4 from [9].

[12] The definition given here is equivalent, and almost identical, to the corresponding definition in [15]. (Recall that we represent interpretations as maximal consistent sets of literals.)

In order to compare formula update and rule update in a precise fashion, we introduce the following additional definition.

Definition 7.2. Given a set $\mathcal{R}$ of inference rules, we define a corresponding set of formulas *Theory*($\mathcal{R}$) as follows:

$$\mathit{Theory}(\mathcal{R}) = \left\{\phi \supset \psi : \frac{\phi}{\psi} \in \mathcal{R}\right\}.$$

Thus, for example, $\mathit{Theory}(\mathcal{R}_1) = \{a \supset \neg b \vee \neg c\}$.

Let $\mathcal{R}$ be a set of inference rules and I an interpretation. Notice that $Cn(I)$ is closed under $\mathcal{R}$ if and only if I is a model of *Theory*($\mathcal{R}$). Thus, every update of I by $\mathcal{R}$ is a model of *Theory*($\mathcal{R}$). In fact, we have the following stronger result, which shows that rule update satisfies the principle of minimal change.

Proposition 7.1. Let $\mathcal{R}$ be a set of inference rules and I an interpretation. Every update of I by $\mathcal{R}$ is a formula update of I by Theory($\mathcal{R}$).

PROOF. Assume that I' is an update of I by $\mathcal{R}$. So I' is a model of *Theory*($\mathcal{R}$). Let I'' be a model of *Theory*($\mathcal{R}$) such that $I' \cap I \subseteq I'' \cap I$. We need to show that $I'' = I'$. Since I' and I'' are both interpretations, it is enough to show that $I' \subseteq I''$.

$$\begin{aligned}
I' &= \{L\colon I \cap I' \vdash_{\mathcal{R}} L\} && (I' \text{ is an update of } I \text{ by } \mathcal{R})\\
&\subseteq \{L\colon I \cap I'' \vdash_{\mathcal{R}} L\} && (I' \cap I \subseteq I'' \cap I)\\
&\subseteq \{L\colon I'' \vdash_{\mathcal{R}} L\} && (I'' \cap I \subseteq I'')\\
&= I'' && (I'' \text{ is a model of } \mathit{Theory}(\mathcal{R})).
\end{aligned}$$ □

The converse of Proposition 7.1 does not hold in general. For instance, we have seen (Example 6.1) that I_3 is not an update of I_1 by $\mathcal{R}_1$, and yet it is easy to verify that I_3 is a formula update of I_1 by *Theory*($\mathcal{R}_1$).

On the other hand, the following proposition shows that if every inference rule in $\mathcal{R}$ has the form

$$\frac{\mathit{True}}{\phi}$$

then the updates of I by $\mathcal{R}$ will be exactly the formula updates of I by *Theory*($\mathcal{R}$). Thus, rule update includes formula update as a simple special case. And since rule update also subsumes revision programming, we see that rule update both generalizes and unifies these two approaches to interpretation update.

Proposition 7.2. Let $\mathcal{R}$ be a set of inference rules, each of which has the form True/ϕ. For any interpretation I, every formula update of I by Theory($\mathcal{R}$) is also an update of I by $\mathcal{R}$.

PROOF. Assume that I' is a formula update of I by *Theory*($\mathcal{R}$). Let I'' be a model of $(I \cap I') \cup \mathit{Theory}(\mathcal{R})$. So I'' is a model of *Theory*($\mathcal{R}$). Also, $I' \cap I \subseteq I''$, so $I' \cap I \subseteq I'' \cap I$. Since no model of *Theory*($\mathcal{R}$) is closer to I than I' is, we can conclude that $I'' = I'$. Thus, I' is the only model of $(I \cap I') \cup \mathit{Theory}(\mathcal{R})$. It follows that $I' = \{L : (I \cap I') \cup \mathit{Theory}(\mathcal{R}) \vdash L\}$. Due to the special form of the rules in $\mathcal{R}$, we see that $(I \cap I') \cup \mathit{Theory}(\mathcal{R}) \vdash \phi$ iff $I \cap I' \vdash_{\mathcal{R}} \phi$ for every formula ϕ. Therefore, $I' = \{L\colon I \cap I' \vdash_{\mathcal{R}} L\}$. □

We may say that Proposition 7.2 shows that "directionality" plays no essential role in inference rules of the form $True/\phi$. To consider another extreme case, the following straightforward proposition shows that rules of the form

$$\frac{\phi}{False}$$

only eliminate updates.

Proposition 7.3. Let $\mathcal{R}$ be a set of inference rules. Let I and I' be interpretations. For any formula ϕ, I' is an update of I by $\mathcal{R} \cup \{\phi/False\}$ if and only if I' is an update of I by $\mathcal{R}$ such that $I' \not\models \phi$.

In fact, we see that if every rule in $\mathcal{R}$ has the form $\phi/False$, then I' is an update of I by $\mathcal{R}$ if and only if I is a model of $Theory(\mathcal{R})$ and $I' = I$. Intuitively speaking, this is the most pronounced example of the effect of the directionality of inference rules. At the other extreme, we have seen that if every rule in $\mathcal{R}$ has the form $True/\phi$, then I' is an update of I by $\mathcal{R}$ if and only if I' is a formula update of I by $Theory(\mathcal{R})$. Thus, in such cases, the directionality of rules has no effect at all. We briefly explore in the remainder of this section the middle ground that lies between these two extremes.

Definition 7.3. Let $\mathcal{R}$, $\mathcal{R}'$ be sets of inference rules. We say $\mathcal{R}'$ is *as strong as* $\mathcal{R}$ if for all sets Γ of formulas, if Γ is closed under $\mathcal{R}'$, then Γ is closed under $\mathcal{R}$.

It is clear that if R' is as strong as $\mathcal{R}$, then for any formula ϕ, $\Gamma \vdash_{\mathcal{R}'} \phi$ whenever $\Gamma \vdash_{\mathcal{R}} \phi$. We use this fact in the proof of the following proposition.

Proposition 7.4. Let $\mathcal{R}$ and $\mathcal{R}'$ be sets of inferences rules such that $Cn(Theory(\mathcal{R})) = Cn(Theory(\mathcal{R}'))$. Let I be an interpretation. If $\mathcal{R}'$ is as strong as $\mathcal{R}$, then every update of I by $\mathcal{R}$ is also an update of I by $\mathcal{R}'$.

PROOF. Assume that $\mathcal{R}'$ is as strong as $\mathcal{R}$, and that I' is an update of I by $\mathcal{R}$. Since $Cn(Theory(\mathcal{R})) = Cn(Theory(\mathcal{R}'))$ and $Cn(I')$ is closed under $\mathcal{R}$, we know that $Cn(I')$ is closed under $\mathcal{R}'$. Consider any $L \in I'$. Since $I \cap I' \vdash_{\mathcal{R}} L$, and since $\mathcal{R}'$ is as strong as $\mathcal{R}$, it follows by previous observation that $I \cap I' \vdash_{\mathcal{R}'} L$. □

Now we define an ordering on inference rules that, intuitively speaking, allows us to compare the degree of directionality in (otherwise similar) rules.

Definition 7.4. Let ϕ, ϕ', ψ, ψ' be propositional formulas.

$$\frac{\phi}{\psi} \preceq \frac{\phi'}{\psi'} \quad \text{iff} \quad \phi \vdash \phi' \quad \text{and} \quad \vdash (\phi \supset \psi) \equiv (\phi' \supset \psi').$$

Example 7.1. By the preceding definition, we have the following:

$$\frac{a \wedge b \wedge c}{False} \preceq \frac{a \wedge b}{\neg c} \preceq \frac{a}{\neg b \vee \neg c} \preceq \frac{True}{\neg a \vee \neg b \vee \neg c}.$$

Roughly speaking, the idea behind this ordering of rules is that, as we move from left to right, the degree of directionality in the rule is lessened, which makes the rule "stronger." Below, we make this claim precise.

Let $\mathcal{R}$ be a set of inferences rules, and let r and r' be inference rules such that $r \preceq r'$. It is clear that $Cn(Theory(\mathcal{R} \cup \{r\})) = Cn(Theory(\mathcal{R} \cup \{r'\}))$. Moreover, it follows easily from the definitions that $\mathcal{R} \cup \{r'\}$ is as strong as $\mathcal{R} \cup \{r\}$. Thus, we have the following corollary to Proposition 7.4.

Corollary 7.1. Let $\mathcal{R}$ be a set of inferences rules, and let r, r' be inference rules such that $r \preceq r'$. Let I be an interpretation. Every update of I by $\mathcal{R} \cup \{r\}$ is also an update of I by $\mathcal{R} \cup \{r'\}$.

8. EMBEDDING RULE UPDATE IN DEFAULT LOGIC

In this section, we specify a very simple embedding of rule update in default logic, using essentially the same inertia rules used in Section 5 to embed revision programming in logic programming. The resulting default theories use normal defaults to encode the commonsense law of inertia.[13]

Definition 8.1 (Translating rule update into default logic). Given a set $\mathcal{R}$ of inference rules and an interpretation I, let

$$\mathcal{D}(\mathcal{R}, I) = \mathcal{R} \cup \left\{ \frac{: L}{L} : L \in I \right\}.$$

Theorem 8.1 (Embedding rule update in default logic). *Let $\mathcal{R}$ be a set of inference rules and I an interpretation. The following hold.*

- *An interpretation I' is an update of I by $\mathcal{R}$ if and only if $Cn(I')$ is an extension of $\mathcal{D}(\mathcal{R}, I)$.*
- *For every consistent extension E of $\mathcal{D}(\mathcal{R}, I)$, there is an interpretation I' such that $E = Cn(I')$.*

PROOF. For part one, let I' be an interpretation. Observe that

$$\mathcal{D}(\mathcal{R}, I)^{Cn(I')} = \mathcal{R} \cup (I \cap I')$$

which justifies the last step below.

$$\begin{aligned} I' \text{ is an update of } I \text{ by } \mathcal{R} \quad & \text{iff} \quad I' = \{L\colon I \cap I' \vdash_{\mathcal{R}} L\} \\ & \text{iff} \quad Cn(I') = \{\phi\colon I \cap I' \vdash_{\mathcal{R}} \phi\} \\ & \text{iff} \quad Cn(I') \text{ is the extension of } \mathcal{R} \cup (I \cap I') \\ & \text{iff} \quad Cn(I') \text{ is an extension of } \mathcal{D}(\mathcal{R}, I). \end{aligned}$$

For part two, assume that E is a consistent extension of $\mathcal{D}(\mathcal{R}, I)$. Suppose there is no interpretation I' such that $E = Cn(I')$. So there is an atom A such that $A \notin E$ and $\neg A \notin E$. But $\mathcal{D}(\mathcal{R}, I)$ includes one of the following two inertia rules:

$$\frac{: A}{A} \quad \frac{: \neg A}{\neg A}.$$

[13] See [13, 14] for applications of essentially this encoding of the commonsense law of inertia in default theories and logic programs for representing knowledge about actions.

It follows that $\mathcal{D}(\mathcal{R}, I)^{\nu}$ includes either A or $\neg A$, and thus E does also. Contradiction. □

9. CONCLUDING REMARKS

Rule update is a simple and expressive framework for interpretation update which extends both revision programming and Winslett's approach to update by means of formulas. Furthermore, it has a simple embedding in default logic, based on a straightforward encoding of the commonsense law of inertia—the principle that things change only when made to.

In [7], Marek and Truszczyński proposed a different extension of revision programming, called *disjunctive revision programming*. Disjunctive revision programs consist of rules of the following form:

$$in(p_1) \vee \cdots \vee in(p_k) \vee out(r_1) \vee \cdots \vee out(r_l) \leftarrow in(q_1), \ldots, in(q_m), out(s_1), \ldots, out(s_n).$$

However, the proposed definition exhibits what may be undesirable behavior. For example, given the disjunctive revision program

$$P = \{in(a) \vee in(b)\}$$

and initial knowledge base $B_I = \{a\}$, we obtain the knowledge base $B_R = \{a,\ b\}$ as one of the two P-justified revisions of B_I, which violates the principle of minimal change.[14] By contrast, the only update of the interpretation $\{a, \neg b\}$ by

$$\mathcal{R} = \{a \vee b\}$$

is $\{a, \neg b\}$ itself. In fact, as noted earlier, Proposition 7.1 shows that rule update never violates the principle of minimal change, although, as we have seen, it is actually based on a principle of "causal" or "justified" change.

Finally, since revision programming is equivalent to logic programming under the stable semantics, computational methods developed for the stable semantics (or, perhaps, for its approximations, such as the well-founded semantics) can be used to provide a query answering mechanism for revision programming. Similarly, since rule update can be embedded in default logic, it should be possible to use computational methods developed for default logic to compute rule update.

Thanks to Vladimir Lifschitz, Norman McCain, and Halina Przymusinska for many helpful discussions and suggestions. We are also grateful for comments from Chitta Baral, Enrico Giunchiglia, and G. N. Kartha. The first author is partially supported by NSF Grant IRI-9313061; the second by NSF Grant IRI-9306751.

REFERENCES

1. Baral, C., Rule-Based Updates on Simple Knowledge Bases, in: *Proc. AAAI-94*, 1994, pp. 136–141.

[14] Mirek Truszczyński has agreed (personal communication) that this example shows that the original definition fails to capture the intended intuition.

2. Gelfond, M. and Lifschitz, V., The Stable Model Semantics for Logic Programming, in: R. Kowalski and K. Bowen (eds.), *Proc. 5th Logic Programming Symp.*, Association for Logic Programming, MIT Press, Cambridge, MA, 1988, pp. 1070–1080.
3. Gelfond, M. and Lifschitz, V., Logic Programs with Classical Negation, in: *Proc. 7th Int. Logic Programming Conf.*, Jerusalem, Israel, Association for Logic Programming, MIT Press, Cambridge, MA, 1990, pp. 579–597.
4. Gelfond, M., Lifschitz, V., Przymusińska, H., and Truszczyński, M., Disjunctive Defaults, in: J. Allen, R. Fikes, and E. Sandewall (eds.), *Principles of Knowledge Representation and Reasoning: Proc. 2nd Int. Conf.*, 1991, pp. 230–237.
5. Katsuno, H. and Mendelzon, A. O., On the Difference Between Updating a Knowledge Base and Revising It, in: J. Allen, R. Fikes, and E. Sandewall (eds.), *Principles of Knowledge Representation and Reasoning: Proc. 2nd Int. Conf.*, 1991, pp. 387–394.
6. Marek, W. and Truszczyński, M., Revision Programming, Research Report, University of Kentucky, 1993.
7. Marek, W. and Truszczyński, M., Revision Specifications by Means of Revision Programs, in: *Logics in AI, Proc. JELIA '94,* Lecture Notes in Artificial Intelligence, Springer-Verlag, 1994.
8. Marek, W. and Truszczyński, M., Revision Programming, Database Updates and Integrity Constraints, in: *Proc. 5th Int. Conf. Database Theory—ICDT 95*, Springer-Verlag, 1995, pp. 368–382.
9. McCain, N. and Turner, H., A Causal Theory of Ramifications and Qualifications, in: *Proc. IJCAI-95*, 1995, pp. 1978–1984.
10. Przymusinski, T. C., Stable Semantics for Disjunctive Programs, *New Generation Computing J.* 9:401–424 (1991). (Extended abstract appeared in: Extended Stable Semantics for Normal and Disjunctive Logic Programs, *Proc. 7th Int. Logic Programming Conf.*, Jerusalem, Israel, MIT Press, 1990, pp. 459–477.)
11. Przymusinski, T. C., Static Semantics for Normal and Disjunctive Logic Programs, *Annals of Math. and Artificial Intell.*, Special Issue on Disjunctive Programs (14):323–357 (1995).
12. Reiter, R., A Logic for Default Rreasoning, *Artificial Intell.* 13(1/2):81–132 (1980).
13. Turner, H., Representing Actions in Default Logic: A Situation Calculus Approach, in: *Working Papers of the 3rd Symp. Logical Formalizations of Commonsense Reasoning*, 1996.
14. Turner, H., Representing Actions in Logic Programs and Default Theories: A Situation Calculus Approach, Unpublished.
15. Winslett, M., Reasoning about Action Using a Possible Models Approach, in: *Proc. AAAI-88*, 1988, pp. 89–93.